

Pivotal.

# Cloud Native Architecture

---

2018.1  
Pivotal

# 클라우드 도입 현실화

# 언론에 비친 클라우드

## ‘클라우드’ IoT 시대 두뇌로

# 클라우드, 이젠 **스타트업 핵심** 아이템

한국 스타트업(창업 초기 기업) 트렌드를 이야기하는 중심에 있는 클라우드 서비스가 있다. 몇 년 전부터 각광받았던 클라우드 기술이 이번 스타트업 혁신 아이템으로 자리 잡고 있다. 이런 흐름 속에서 클라우드 서비스를 비단으로 업계 주목을 받고 있는 스타트업 두 곳에 만날었다.

클라우드 서버 '수루산' 업체 ASD네트워크로  
클라우드 서버  
비음과 인력  
본선사나 IT “고”

ASD테크놀로지가 유통  
기업을 위한 클라우드 서비스를  
제작하면 그 기업은 고유 브랜드  
를 입혀 그 회사 클라우드 서비스  
를 제공할 수 있다. 클라우드 솔  
루션을 기업에 제공해 그 기업이  
고객에게 개인용 클라우드 서비  
스를 제공하는 Business To Customer  
사업이다.

고 있다. ASP데브로드로는 있고 있는 ASP데브로도지는 40여만 130만달러 매출을 올고 올해는 200만달러 이상 수익을 얻을 수 있을 것으로 기대하고 있다.

클라우드 오피스 개발업계 '구글다스'도 뛰어난 기술력을 바탕으로 회사의 주목을 받고 있는 스트리트업이다. 한글과컴퓨터(한국개발자 출신들이 창업 창업) '구글다스'는 구글다스, 마이크

우모해 보이는 도전이  
스는 기술력만큼은 기  
입장이다.  
구글드스를 이용할  
서서인 워드, 파워포인  
거나 도표 등이 깨지는  
가 생겼다. 루루드스는  
업률도 제대로 해결하  
서 간호한 문제를 원  
하는 데 채 1년도 걸리  
4. 주문

## SW업계, 클라우드에 올인

\* 출처: 지능-정보-사회와-클라우드 클라우드혁신센터

# 이슈를 넘어 현실화

## 전 세계 IT산업은 클라우드컴퓨팅 방식으로 IT 패러다임 변화

전 세계 퍼블릭 클라우드 서비스 시장 규모  
2020까지 약 2배로 급성장

IDC 2016년 예측



- 국내 클라우드 시장은 '16년 1.19조원이며, '18년은 1.97조원 예상

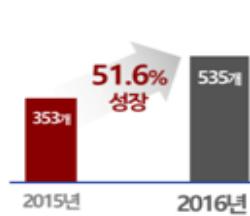
2016년 국내 클라우드 시장규모  
약 1.19조원 | 전년대비 55% 성장

실태조사, NIPA

### 클라우드 시장 규모



### 국내 클라우드 기업 수



글로벌기업, 제조업, SW에 걸쳐  
다양한 분야에 클라우드 확산



### 해외 클라우드 동향

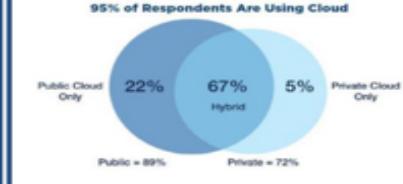
국가	동향	설명
미국	다양한 산업 적용	<ul style="list-style-type: none"><li>아마존, MS, 세일즈포스닷컴 등 기준 강자는 IoT/빅데이터를 결합하여 제조업 등에 적용 중</li><li>※ 클라우드에 IoT를 적용할 수 있는 플랫폼으로 확대</li></ul>
중국	산업 급성장	<ul style="list-style-type: none"><li>정부의 산업 육성 의지, 바이두 /알리바바/텐센트 등 적극적 투자로 연평균 50% 성장 중</li><li>※ '15년 시장규모 : 16.3조원(중국 청과연구센터)</li></ul>
일본	CSB 활성화	<ul style="list-style-type: none"><li>글로벌 기업과 자국의 SI 및 클라우드 기업이 공존하는 CSB 시장이 활성화 중</li><li>※ 시장규모(IDC재팬, '15년) : '14년 3.5조 → '19년 10.1조</li></ul>
기타	산업의 기반기술	<ul style="list-style-type: none"><li>글로벌 제조사인 GE는 산업용 클라우드 개발*</li><li>구글(알파고), IBM(왓슨)은 클라우드로 AI 구현</li></ul>

\* 출처: 지능-정보-사회와-클라우드\_클라우드혁신센터

# 클라우드 주요 현상

웹서비스 중심에서 모든 산업으로, 신규 구축에서 레거시 이동으로 클라우드 도입

## 클라우드 주요 현상

과거	웹서비스 · 게임 · 미디어  Elastic 인프라 환경 Time to Market 실현	기존업체보다 스타트업  클라우드 장점의 극대화	레거시보다 신규구축  전환비용 없고, 리스크가 없음	국내서비스보다 글로벌서비스  글로벌 서비스는 레거시 방식의 가장 큰 혜택
현재	법규제개선  금융, 의료, 교육 등의 법규제개선	공공 · 의료 · 금융분야 도입  공공기관 클라우드 도입 활성화	보안을위해도입  보다 강화된 보안과 시스템 안정성을 위해 클라우드 전환	하이브리드 구축  95% of Respondents Are Using Cloud Public Cloud Only: 22% Hybrid: 67% Private Cloud Only: 5% Public = 89% Private = 72% 기존 레거시가 클라우드로 전환

# 클라우드 사용 이유

Cloud Benefits 2017 vs. 2016

% of Respondents Reporting These Benefits



Source : Rightscale 2017 State of the Cloud Report



비용절감  
인력효율

---

# Why Cloud Native?

# Scale and Resilience

모든 것은 항상  
실패한다!

---

It's not Amazon's Fault

# AWS suffers a five-hour outage in the US

21 September 2015 | By Max Smolaks



Ten services hosted in the Northern Virginia data center experience disruption

**28/02/2017 update: for information on the more recent AWS outage, [CLICK HERE.](#)**

Cloud provider Amazon Web Services (AWS) suffered an outage on Sunday morning that sent shockwaves through the Web, with some users being unable to reach popular online services like Netflix, Tinder, Airbnb, Reddit and IMDb offline.

# 탄력적인 확장

---

자동/수동 수평 확장/축소

- 구성 변경이 잦음

- 실행 앱 인스턴스(VM/Container) 개수 탄력적 변경

- LB에 추가 인스턴스 등록/삭제?
- API 호출 시 서비스 서버 목록 관리는?
- 모니터링은?

- 시스템의 구성 정보(IP, Hostname, ...) 동적 변경

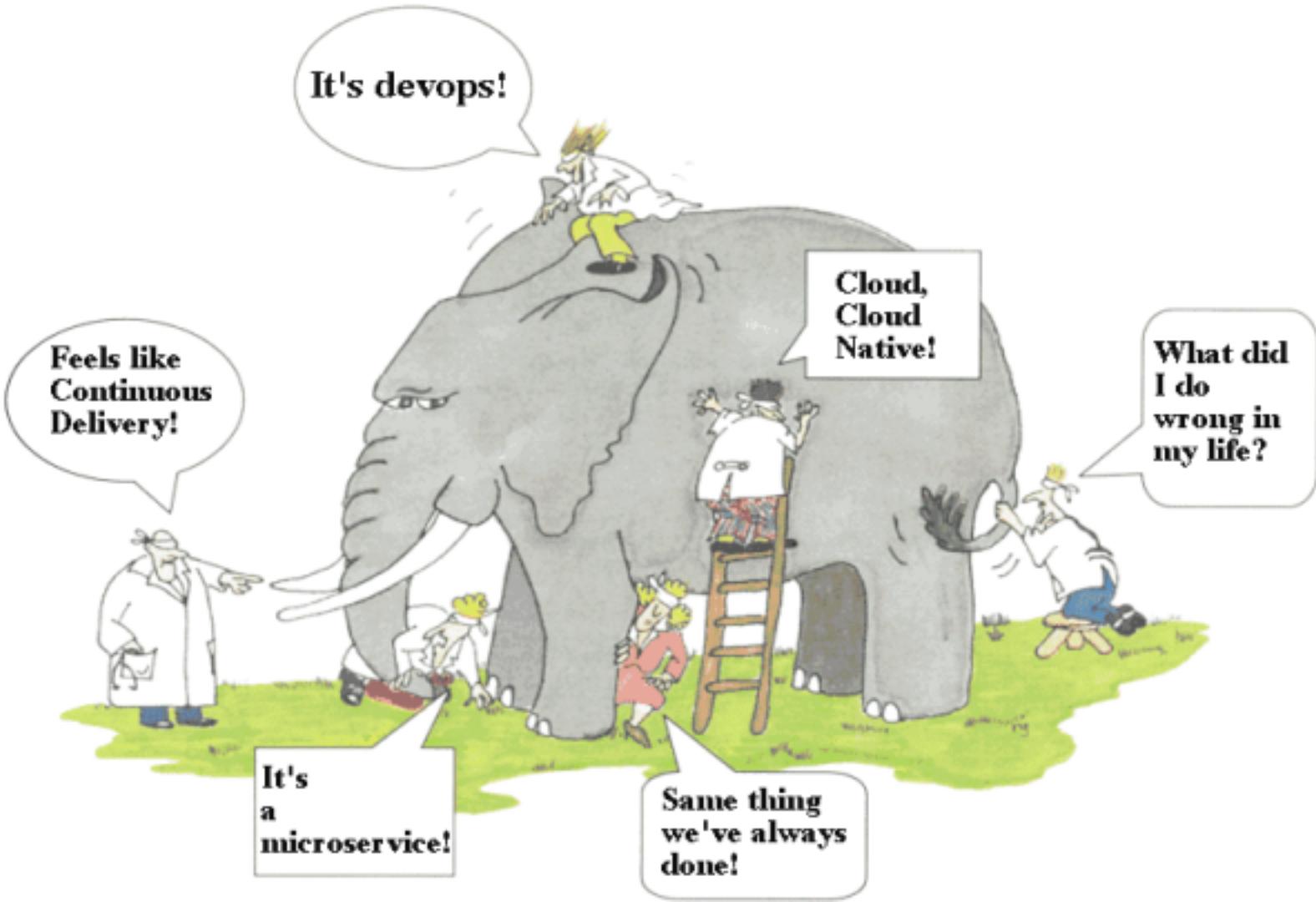
- 어떻게 서비스를 찾을 것인가?
- 기존 모니터링 시스템에서 관리가 가능한가?

- Multi-Cloud 고려

- Hybrid Cloud 또는 Multi Vendor Cloud 구조 확장시?

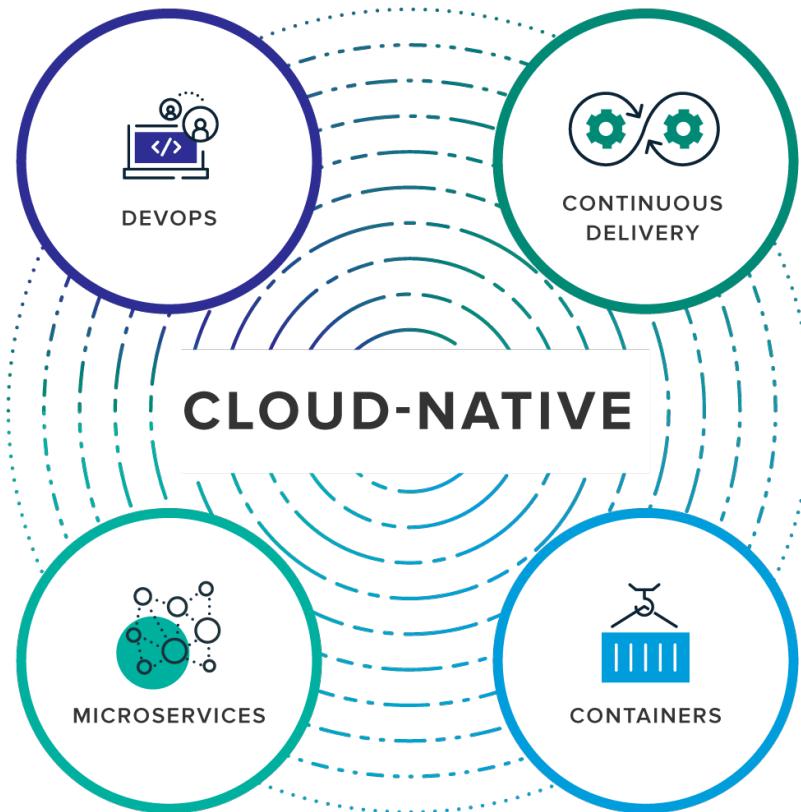
---

# What is Cloud Native Architecture?



클라우드 컴퓨팅 모델의 장점을 모두  
활용하는 애플리케이션을 개발하고  
실행하기 위한 접근 방식

# Cloud Native 핵심 요소



---

# 지속적인 서비스 딜리버리 ( Continuous Delivery )



A Seattle book store  
deploys code, on average,  
every second

# Software Cycles Have Been Getting Faster

Infrequent (Manual) Deployments



10 Deploys/Day

(2009)  
**flickr**



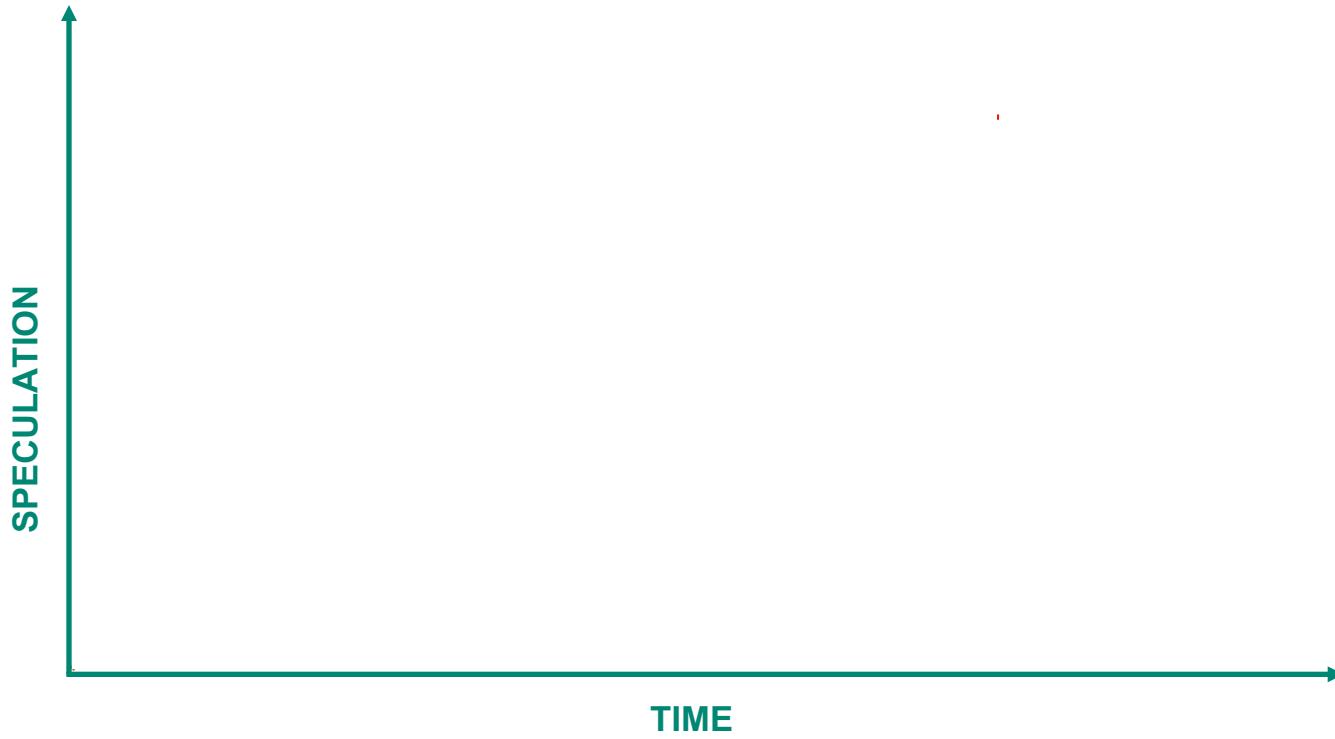
100 Deploys/Day

(2013)  
**NETFLIX**



Continuous (Automated) Delivery

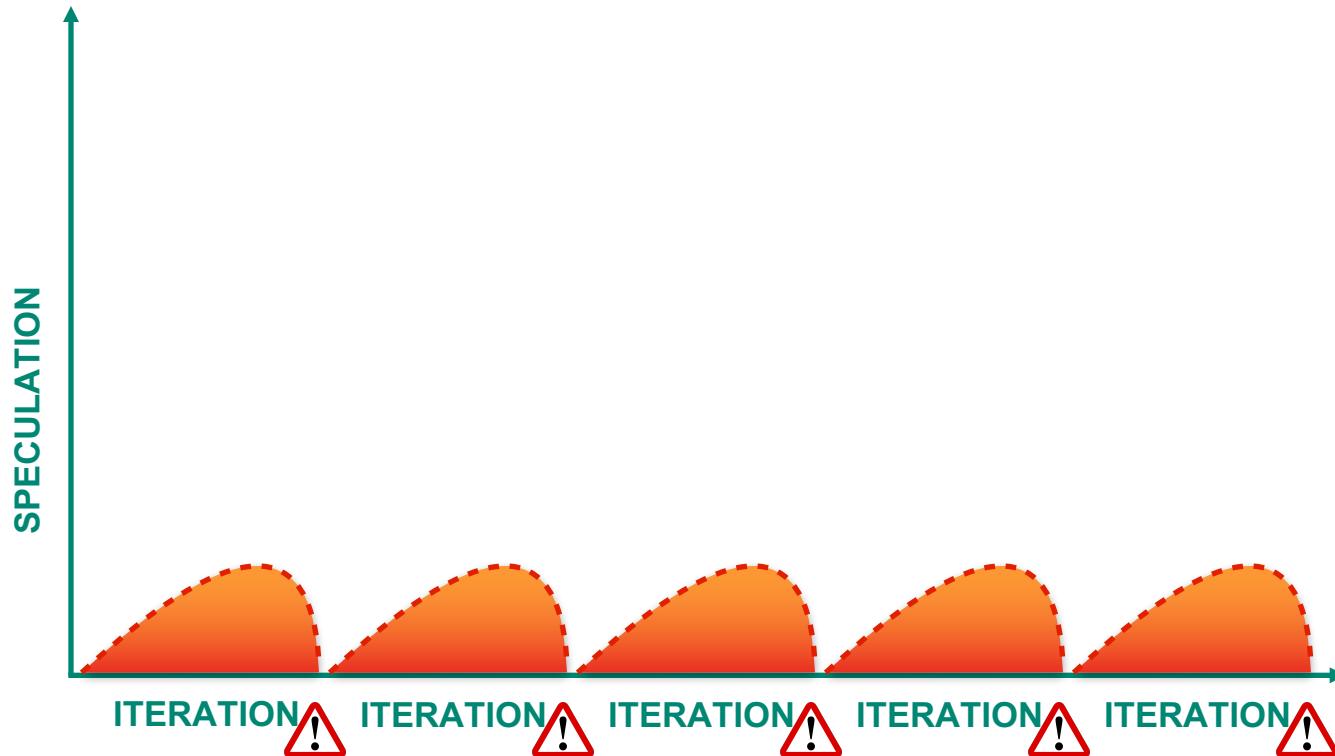
# Why?



# Risk Accumulation



# Experimentation Reduces Risk



# **Continuous Delivery**

**of**

# **Customer Value**

# Obstacles

- Silos: Dev, QA, Operations is typical.  
No shared common goal
- Dissimilar Environments - “It works  
on my machine”
- Risky Deployments: Manual steps,  
done “off hours”
- Changes are treated as an  
exception, not the norm  
→ Firefighting
- Processes designed around these  
obstacles



# 7 Prerequisites for Continuous Delivery

---

1. DevOps culture
2. Continuous Integration
3. Automate and version tasks, for auditing & reproduction
4. Shared tools & procedures
5. Apps must be easy to deploy to prod & easy to rollback
6. App must be deployed to programmable target
7. Application versions must be ready to be shipped into production

# Continuous Integration & Delivery

Deliver High Quality Software Faster & Continuously,  
from Idea to Production

**AUTOMATION.**  
Integrate tools and automate processes from testing to builds and deployment

**SPEED.**  
Release more frequently with smaller bits will reduce complexity and improve time-to-market

**QUALITY.**  
Reduce feedback loop using test-driven development to surface problems sooner and be responsive

**AGILITY.**  
Push updates on regular basis with no downtime to improve customer experience and time to market

Build Pipeline Operations



Tool Chain



GitLab



Concourse



jFrog Artifactory



Development



Test + UAT + Staging



Production

Distributed revision control and source code management.  
Collaborative software development

Build and test software projects continuously and incrementally.  
Hundreds of compatible plugins

Share binaries and manage distributions.  
Manage artifact lifecycle. Avoid license violations

Develop, test, QA and production on the same platform. Simple, developer friendly commands and APIs. Operational benefits for every app. Built-in ecosystem services. Deploy, operate and scale on any IAAS

---

# 데브 옵스 ( DevOps )

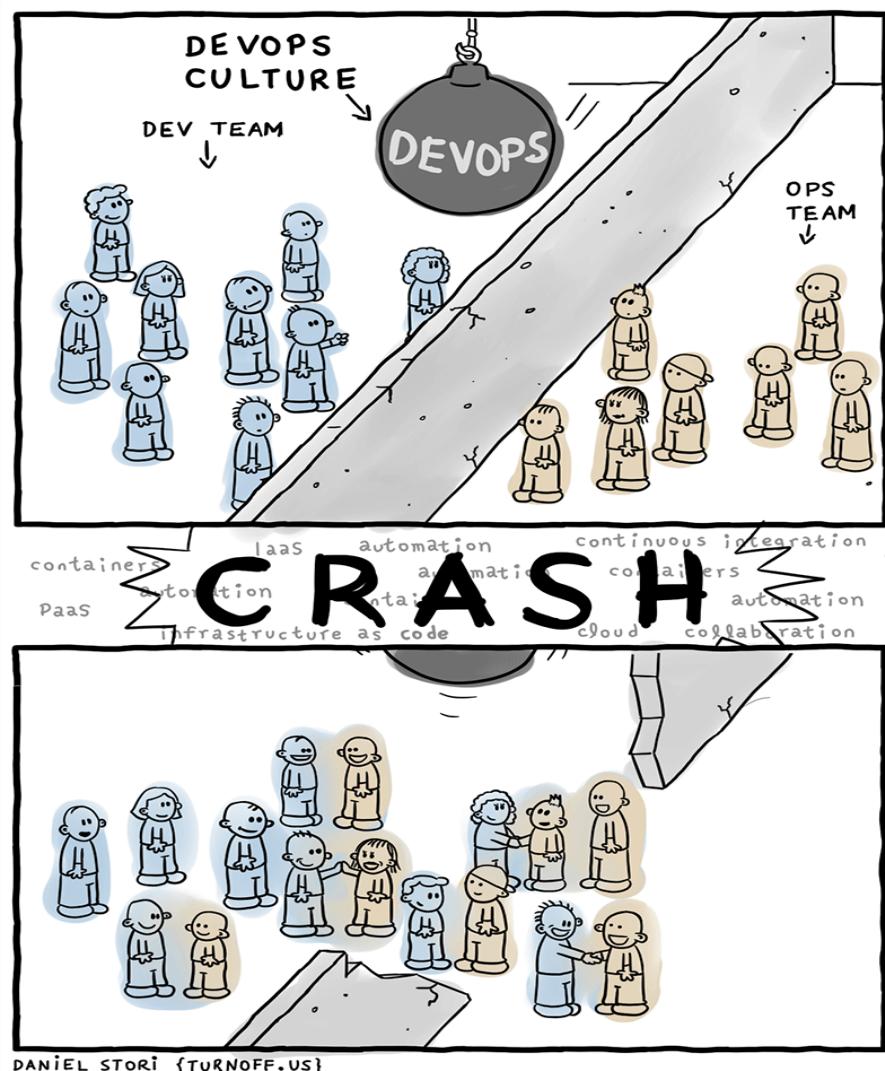
# The History of IT Organizational Systems

- 1960-1980s: ***Fragile*** “**Cowboys**”
    - Optimize around the **hardware**
  - 1990-2000s: ***Robust*** “**ITIL**”
    - Optimize and protect around **changes** or **traumas**
  - 2010s+: ***Resilient/Anti-Fragile*** “**DevOps**”
    - Optimize on continuously improving the **flow of value**

# DevOps?

- DevOps is a way of working that focuses on regularly shipping quality software that meets a business need ...

- DevOps is fluid, and avoids much of the traditional handoff friction and delays between product development and IT operations through ...



# 피자 두판의 팀이란?

A photograph of two pizzas placed side-by-side on a light-colored wooden surface. The pizza on the left is topped with melted cheese, black olives, and a sprig of parsley. The pizza on the right is topped with pepperoni, melted cheese, and finely chopped green herbs. Both pizzas have a golden-brown crust.

각 팀이 분산된  
민첩하면서, 독립적인,  
신뢰하고, 오너쉽을  
가진 서비스 팀

**“DevOps”**

# Example Platform Team: Pivotal CloudOps

~6000+ Application Servers: 7 Full Time Staff and 1 Rotation



---

# 마이크로 서비스 ( Micro Services )

July 15th is Prime Day

[Learn More](#)



Shop by  
Department

All ▾ 11 seconds

Search  μservice



Shop Beautiful Things, Updated Daily

Your Amazon.com

Today's Deals

Gift Cards

Sell

Help

Hello, Sign in  
Your Account

Try  
Prime

Wish  
List 0  
Shopping  
Cart

Books

Advanced Search

New Releases

Best Sellers

The New York Times® Best Sellers

Children's Books

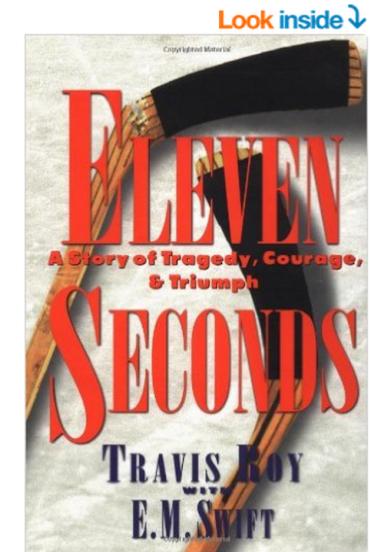
Textbooks

Textbook Rentals

Sell Us Your Books

Best Books of the Month

Deals in Books



Look inside ↴

Copyright Material



See all 3 images

Image μservice

## Eleven Seconds: A Story of Tragedy, Courage & Triumph

Hardcover – January 1, 1998

by [Travis Roy](#) ▾ (Author), [E. M. Swift](#) (Author)

62 customer reviews

#1 Best Seller in Hockey Biographies

Item Master μservice

Reviews μservice

See all 4 formats and editions

Kindle

\$8.05

Hardcover

\$16.85

Read with Our Free App

118 Used from \$0.01

48 New from \$3.95

14 Collectible from \$4.95

In this heartfelt testament to the power of love and the strength of the human spirit, Travis Roy, who suffered a devastating injury eleven seconds into his first college hockey game, reveals how he has managed to cope after the accident and, with the help of family and friends, overcome tremendous barriers to begin a new life.

Best of the Month

See the Best Books of the Month

Want to know our Editors' picks for the best books of the month? Browse [Best Books Month](#), featuring our favorite new books in more than a dozen categories.

Other  
dependent  
μservice

July 10? Order within 3  
Choose Two-Day  
Shipping. Details

Address: ▾

Share



μservice

\$16.85

List Price: \$26.00

Save: \$9.15 (35%)

FREE Shipping on orders over  
\$35.

Only 18 left in stock (more on  
the way).

Ships from and sold by Amazon.com.  
Gift-wrap available.

Yes, I want FREE Two-Day  
Shipping with Amazon Prime



Add to Cart

Turn on 1 Click ordering for this browser

# Microservices

If services have to be updated together,  
they're not loosely coupled!

## Loosely coupled service oriented architecture with bounded contexts

- Adrian Cockcroft

If you have to know about surrounding services,  
you don't have a bounded context

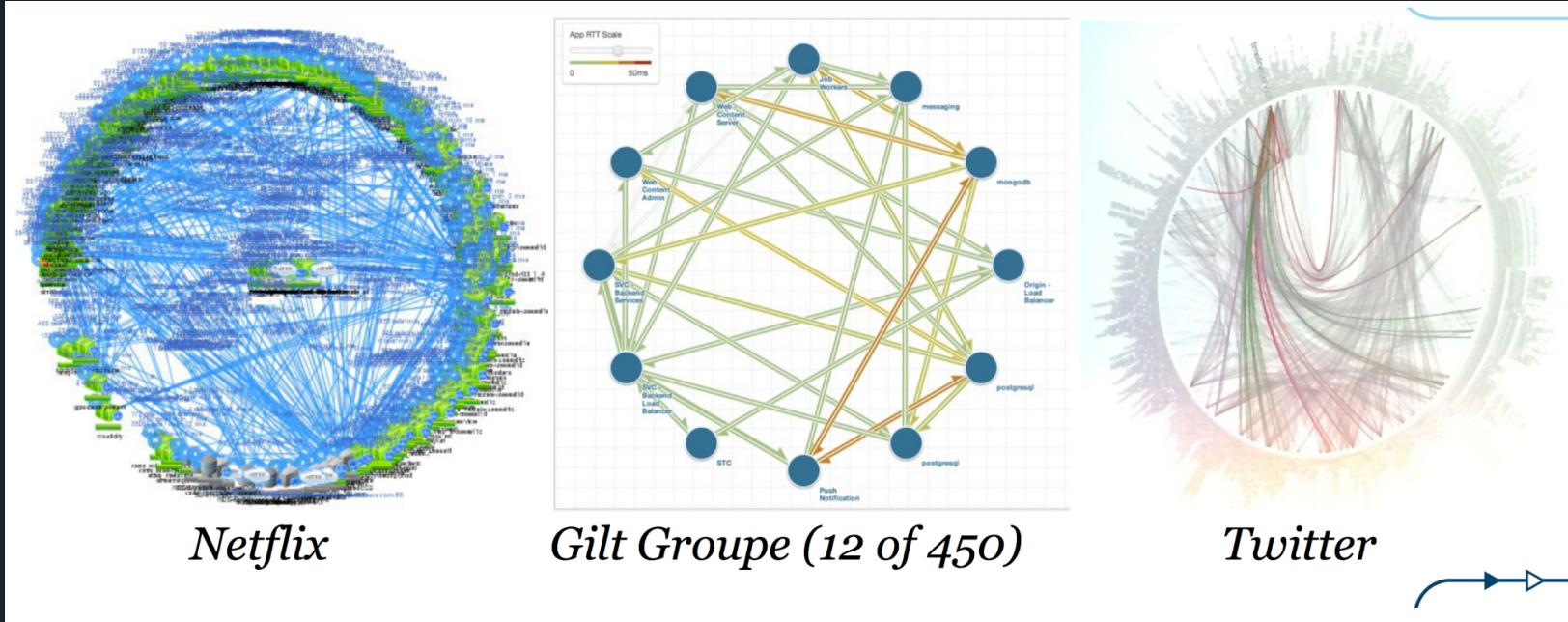
“Bounded Context는 독립적으로 서비스 될 때 문제없는 업무 범위”

# Microservices

---

“각각의 기능별로 독립적인 서비스로  
구성하여 작은 서비스의 집합으로 원하는  
기능을 만들어 내는것”

# Microservice Pioneers



Pivotal™

# Who Moved My Complexity?

---

- Microservices are individually simple
- Complexity is transferred to the ecosystem

# Challenges in a Distributed System

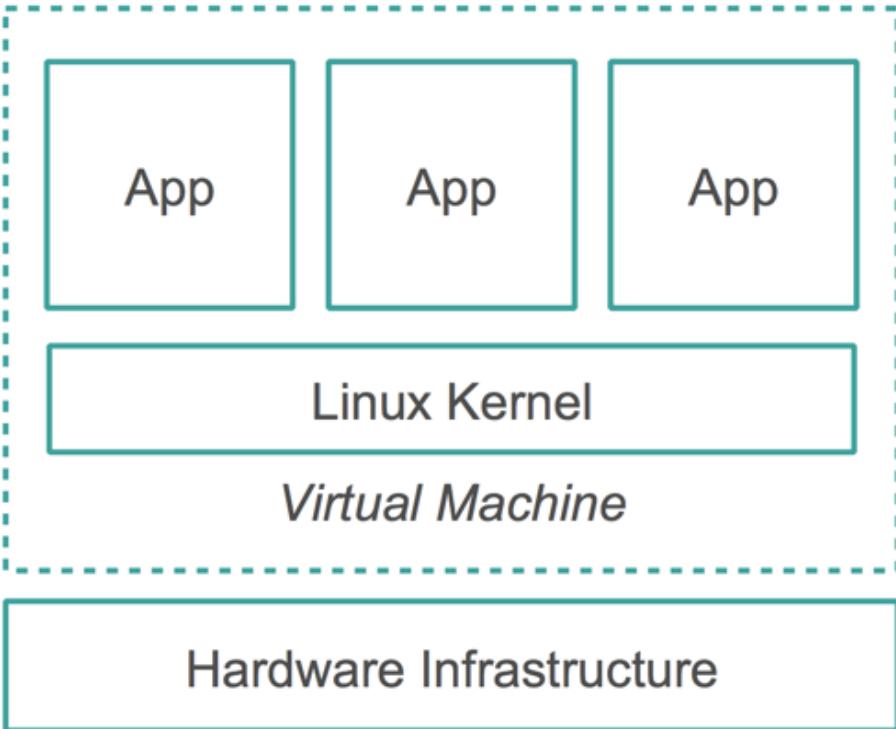
---

- Configuration management
- Registration and discovery
- Routing and load balancing
- Fault tolerance and isolation
- Aggregation and transformation
- Monitoring and distributed tracing
- Process management

---

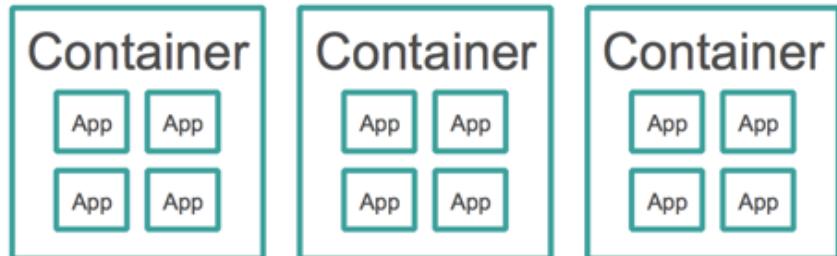
# 컨테이너 (Container)

# Application Server Deployment - Monolith



- Load balancing requires provisioning of new VMs and app server installations
- Poor resource isolation; memory leaks can cause other applications to become unavailable
- Runtime environment is driven by the operator

# Linux Container Deployment - Microservice



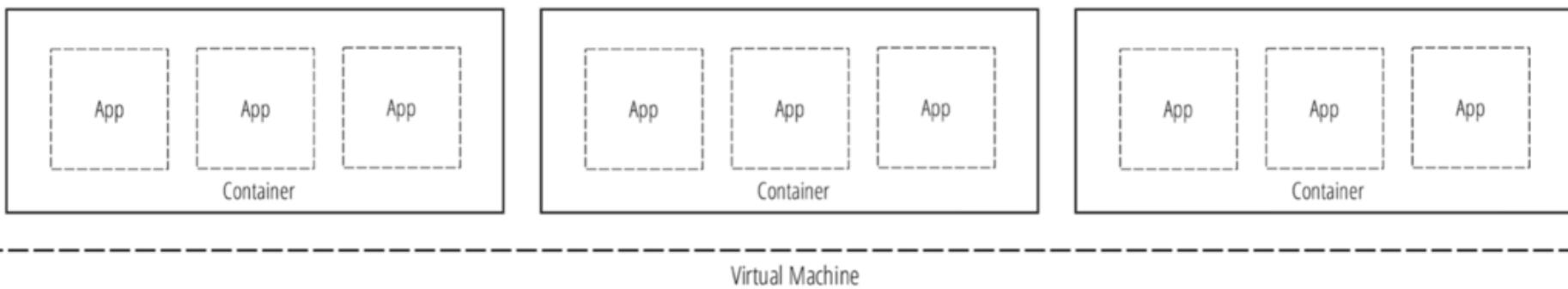
*Virtual Machine*

Hardware Infrastructure

- Development team drives the application runtime of a container
- Containers are resource isolated, allowing efficient scheduling onto a grid of VMs
- Containers take seconds to start, VMs take minutes

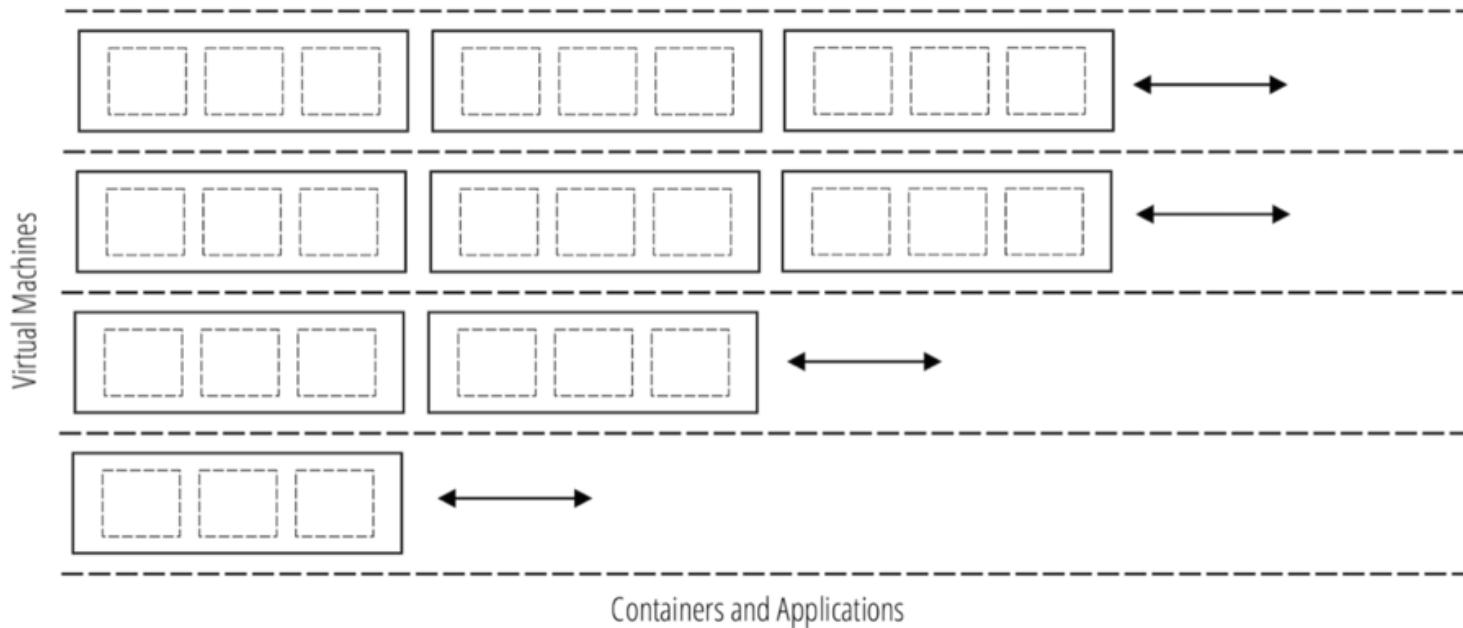
# Microservices - Container Deployment

- Each microservice can be containerized with their application dependencies
- Containers get scheduled on virtual machines with an allotted resource policy



# Container scheduling and auto-scaling

- Minutes to start a VM, but seconds to start a container
- An elastic runtime can perform auto-scaling of containers in PaaS layer



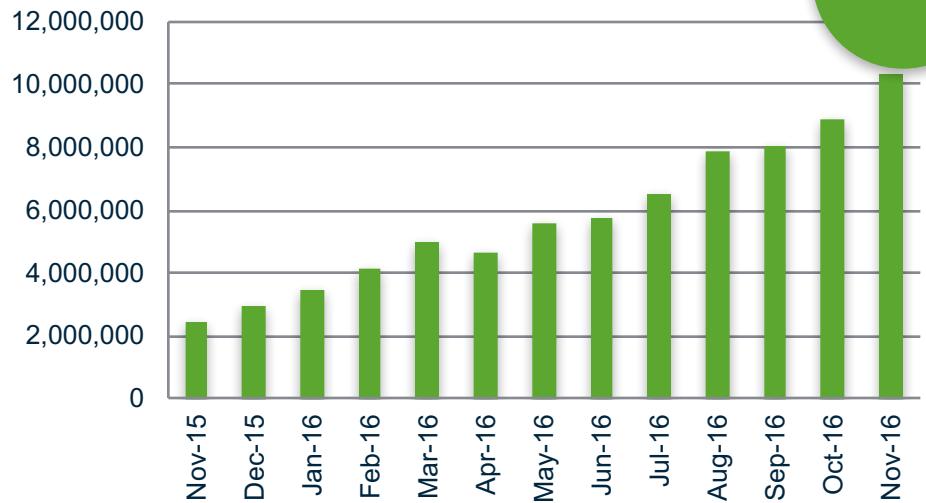
---

# How?

- Tools
- Platform
- Process and Culture

# Spring Boot Popularity

## Monthly Maven Downloads



source: oss.sonatype.org

 Ed Young @summitbid 

Killer marathon presentation @comcast by the @pivotal team. @starbuxman worked some mind blowing black magic for us w #springboot et al

RETWEETS 11 LIKES 10 

9:12 PM - 5 Aug 2015

 Pierre Pureur @PGP60 

Our first #SpringBoot app is in production on @pivotalcf with 30% less code than the JBOSS version - Big thanks to @starbuxman & @pivotal!

RETWEETS 49 LIKES 55 

9:57 AM - 23 Apr 2016

 James Watters @wattersjames - 15 Sep 2015 This morning in Germany a car company launched their connected car product on @pivotalcf and @springboot



 Andrew Glover @aglover 

Spring Boot is the new cool & come learn about @springboot with @phillip\_webb on Tuesday, 9/23 at @netflix!

 Silicon Valley Groovy/Grails Centro Spring Boot, the new convention-over-configuration centric framework from the Spring team at Pivotal, marries Spring's flexibility with conventional, common sense defaults to make ap... meetup.com

RETWEETS 15 LIKES 5 

4:12 PM - 12 Sep 2014

 Rohit Sood @techlogix 

Working with @pivotalcf and breaking down monoliths! #springboot

RETWEETS 1 LIKES 4 

8:19 AM - 31 Aug 2016

# Spring Cloud – Microservices Infrastructure

## Common Patterns for Distributed, Cloud Based Enterprise Applications

Designed for disposable infrastructure in  
partnership with **Netflix**



# Spring Cloud Data Flow – Data Microservices

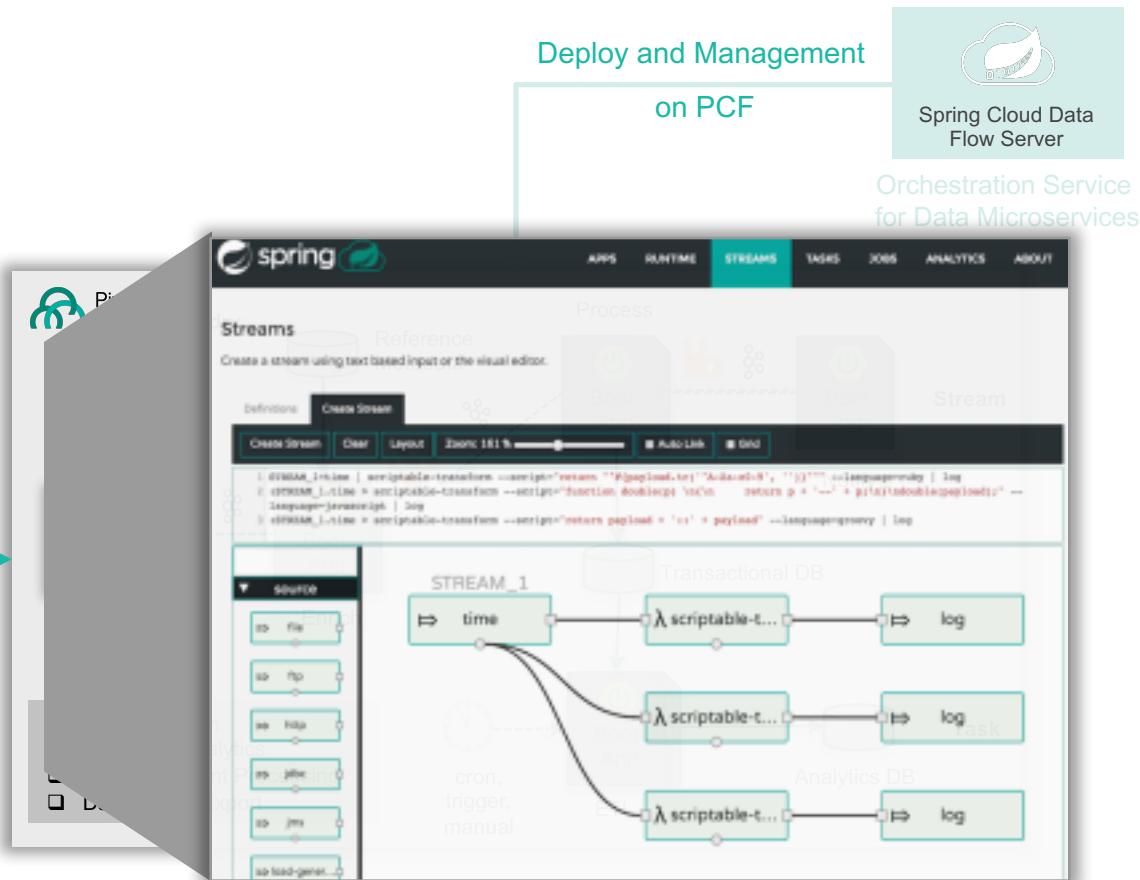
## Developer Benefits

- ✓ Simple programming model
- ✓ Composition: Streaming and Batch
- ✓ Readily Available Building Blocks
- ✓ Messaging Middleware
- ✓ CI/CD + TDD Friendly

## Operator Benefits

- ✓ Canary/rolling upgrades
- ✓ Dynamic failover and auto scaling
- ✓ Easy Failure Isolation
- ✓ Operational Dashboard

HTTP

# Spring Cloud Services

## Microservices Management

The diagram illustrates the architecture and management of a microservices system. At the top, a stack of three layers represents the system environment: Development, QA, and Production. The Production layer contains three service instances: Boot, Order, and Catalog. Two instances are green (healthy), while one is red with a crossed-out power icon, indicating it is disabled or failed.

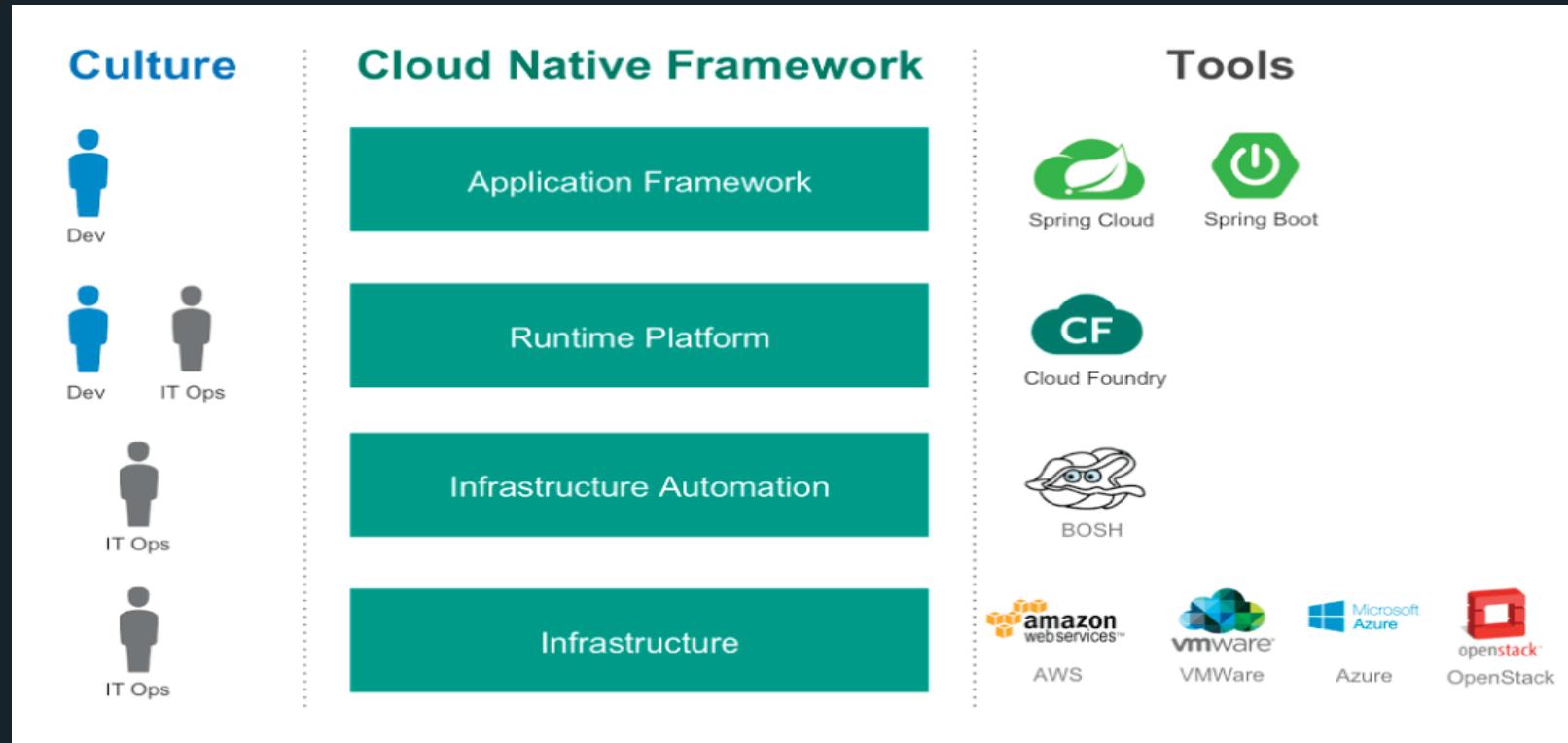
A red dashed line connects the Production layer to a central component: the **Circuit Breaker Dashboard**. This dashboard is shown in a 3D perspective, displaying metrics for three specific services:

- customer-portal.placeOrder**: Host: 0.0/s, Cluster: 0.0/s, Circuit Closed. Metrics: Hosts 0, Median 2ms, Mean 408ms, 1 1ms, 90th 814ms, 99th 99.5th 814ms.
- customer-portal.myOrders**: Host: 0.0/s, Cluster: 0.0/s, Circuit Closed. Metrics: Hosts 0, Median 2ms, Mean 137ms, 1 1ms, 90th 595ms, 99th 595ms.
- customer-portal.menu**: Host: 0.0/s, Cluster: 0.0/s, Circuit Closed. Metrics: Hosts 0, Median 43ms, Mean 48ms, 1 1ms, 90th 53ms, 99th 53ms.

Below the dashboard, a section titled **Thread Pools** shows the state of the **CustomerController** pool: Host: 0.0/s, Cluster: 0.0/s. Metrics: Active 0, Queued 0, Pool Size 5, Max Active 0, Executions 0, Queue Size 5.

On the left side of the dashboard, there are two teal-colored boxes: one labeled "git" with a red diamond icon and another labeled "Co". On the right side, a teal box contains the text "circuit breaker dashboard". A large grey arrow points from the Production layer towards the dashboard, indicating its role in monitoring and managing the system's health.

# Pivotal Cloud Foundry



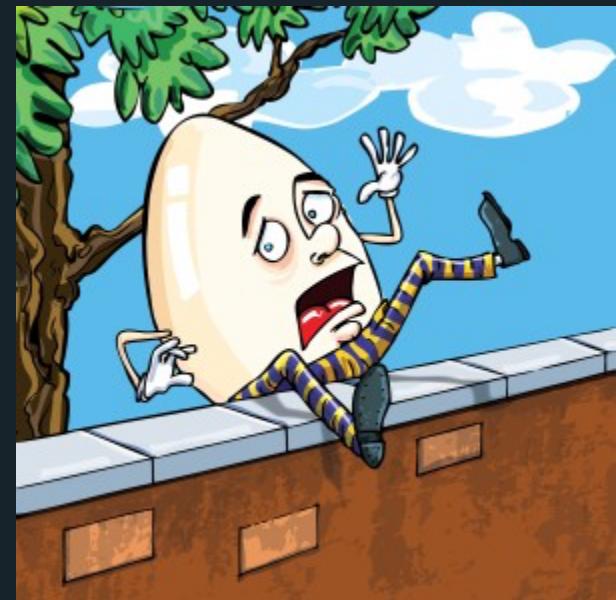
# The Journey: Technology

---

- Provide the right tools
  - Platform, design pattern foundations, continuous delivery...
- Start small
  - Right-size for your needs
  - For migrations, don't boil the ocean
- Target applications that:
  - Have high business/competitive impact
  - Need to scale or change frequently

# The Journey: Culture & Organization

- Small, cross-functional teams
- Products, not projects
- Test-driven development
- Short iterations
- Transparency and visibility





Transforming How The World Builds Software